How to connect to Denodo from Octave

Revision 20211117

Denodo Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

CONTENTS

1 GOAL

GNU Octave is a software primarily intended for numerical computations. It is free software that helps in solving linear and nonlinear problems numerically, and to perform other numerical experiments using a language that is mostly compatible with MATLAB.

This document describes how to connect Octave to Denodo.

2 PREREQUISITES

The following XML Toolbox is required: XMLTree: an XML toolbox for MATLAB/Octave.

3 GETTING INFORMATION FROM DENODO

The following sections show how to read the data from Denodo in Octave using two methods: **HTTP** and **JDBC**.

Through **HTTP** we will connect to a REST Web Service published in Denodo, retrieve the information in XML format and then process it.

Through **JDBC** we will connect directly to a Denodo virtual database and retrieve the information through a SQL Query.

3.1 ACCESSING DENODO THROUGH HTTP

Denodo gives the possibility of consuming data through the creation of both SOAP and REST Web services. In order to get the data, Web Services allow access via HTTP. Since Octave provides functions to perform HTTP access, it will be possible to consume data coming from Denodo Web Services. In the following example we will explain how to make these calls to read the data from a REST Web Service published in Denodo.

1. First of all, execute the command webread with or without parameters to get the data in **XML form** (this command accepts authentication methods with `weboptions`).

   Example without parameters:

   ```
   xmlDoc = webread(urlRestWebService);
   ```

   Example with parameters:

   ```
   xmlDoc = webread(urlRestWebService,parameter1,value1,
   parameter2,value2);
   ```

   Example with authentication:

   ```
   xmlDoc = webread(urlRestWebService,
   weboptions('Username', 'user', 'Password', 'user''));
   ```

2. Second, this retrieved data needs to be transformed from **XML to cells** with these commands.

   - Transform the XML Document in a tree with the XML Toolbox.
     tree = xmltree(xmlDoc**);**
   - This tree can be then transformed into a structure.
     s = convert(tree**);**
   - Finally this structure is converted to a cell so that it is easy to access.
     c = struct2cell(s**);**

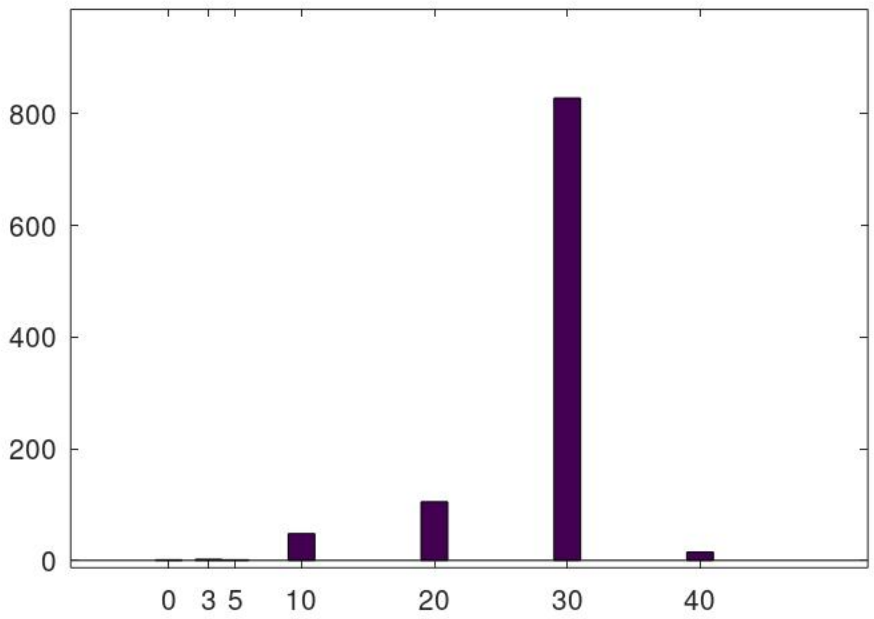
A small example of how to implement this first method.

1. This code in Octave firstly uses the command **`webread`** to retrieve the data from

the REST Web services and then transform it with the following commands: **xmltree, convert** and **struct2cell**. Afterwards it initializes the variables and executes the loop to save the query's information in arrays. Finally it plots the histogram with the command **hist**.

```
clear;
xmlDoc = webread('http://localhost:9090/server/denodo_training/
ws_traffic_crashes/views/f_traffic_crashes');
tree = xmltree(xmlDoc);
s = convert(tree);
c = struct2cell(s);
numRows = size(c{1,1},2);
M = zeros(1,1000);
velM = zeros(1,7);
previous = 600000;
x = 0;
for rowIdx = 1:numRows
  vel = c{1,1}{1,rowIdx}.posted_speed_limit;
  if not(vel == previous)
    x = x+1;
    velM(x) = str2num(vel);
  endif
  M(rowIdx) = str2num(vel);
  previous = vel;
endfor

hist(M,velM)
```

2. The previous code displays the following histogram in Octave:



3.2 ACCESSING DENODO THROUGH JDBC

Denodo provides a Java API that allows executing queries on the Denodo virtual databases. In addition, it also provides a JDBC driver that implements the main characteristics of the JDBC API.

The Denodo JDBC driver can be found either in the Denodo Community or from the Denodo installation folder (`<DENODO_HOME>/tools/client-drivers/jdbc/vdp-jdbcdriver-core/denodo-vdp-jdbcdriver.jar`).

Octave does not have native methods to access a database through JDBC, but it does offer the option of importing Java classes.

The following example shows how to access Denodo by importing the JDBC driver and a custom Java class in Octave:

1. Create a class in java to connect to a Denodo database and return an array of data with a method in the class. Export this class to a jar file.

2. Import the class created and the Denodo VDP JDBC driver to **Octave** with the command **javaaddpath**.

   - **`javaaddpath(`**`path/class.jar`**`);`**
   - **`javaaddpath(`**`path/denodo-vdp-jdbcdriver.jar`**`);`**

3. Get the Java object from the class you did import with the command **javaObject**.

   - **`Object = javaObject (`**`com.example.App`**`);`**

4. Execute the method created in the class accessing through the object created previously

   - **`Object.connection(`**`parameters`**`);`**

Here we have a small explanation about the most important parts of the Java class.

This is the method's signature that connects to the Denodo database and executes the query returning the data in int array form so that Octave can use it:

```
public static int[] connection(String database, String user, String password, String sqlQuery)
```

Firstly, the method connection needs to import the class of the Denodo VDP JDBC Driver for the connection with the database:

```
Class.forName("com.denodo.vdp.jdbc.Driver");
```

Now the class will be able to access the database and execute a query.

```
connection = DriverManager.getConnection(database, user, password);

// Create the statement from the user query
PreparedStatement statement = connection.prepareStatement(vqlQuery);

// Execute the query
statement.execute();

// Obtaining the result of the query and printing it.
ResultSet result = statement.getResultSet();
```

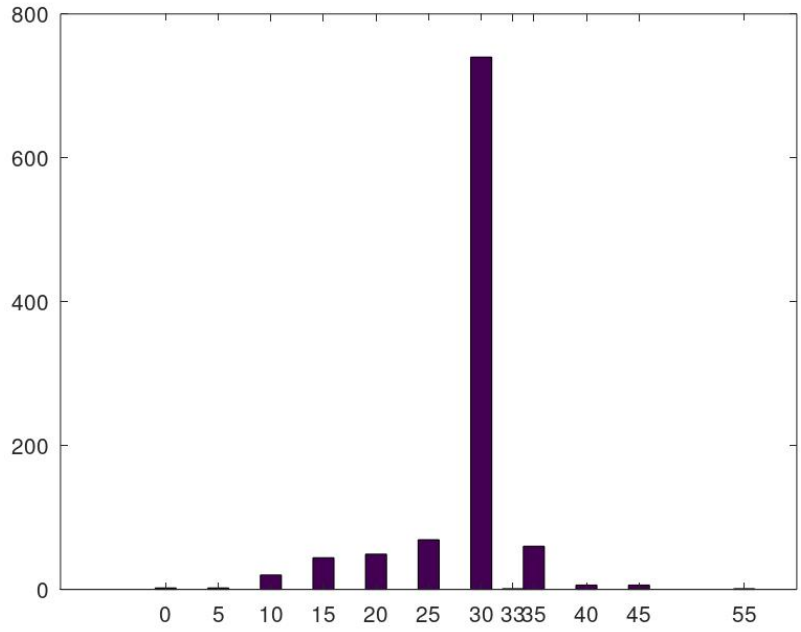
An example of how to implement this second method:

1. The following code imports the Denodo driver with the **javaaddpath** command, creates the object with **javaObject** and then executes the query with the method in this object (connection.connection). Afterwards it initializes the variables and executes the loop to save the query's information in arrays. Finally, it plots the histogram with the command **hist**.

```
clear;
javaaddpath("C:/ConnectionJDBCOctave/javaclass/demo/demo.jar");
javaaddpath("<Denodo_Home>/tools/client-drivers/jdbc/denodo-
vdp-jdbcdriver.jar");
connection = javaObject ("com.example.App");
arrayLimitVel                                                    =
connection.connection("jdbc:vdb://localhost:9999/denodo_training
", "admin", "admin",
"SELECT posted_speed_limit FROM f_traffic_crashes");

numRows = size(arrayLimitVel);
velM = zeros(1,6)
previous = 600000;
x = 0;
rowIdx=1;
arrayLimitVel = sort(arrayLimitVel);

for rowIdx = 1:numRows(1)
  vel = arrayLimitVel(rowIdx);

  if not(vel == anterior)
    x = x+1;
    velM(x) = vel;
  endif

  previous = vel;
```

```
    endfor

    hist(arrayLimitVel ,velM)
```

2. The previous code displays the following histogram in Octave:



4 REFERENCES

WWW Access (GNU Octave (version 6.1.0))
GNU Octave: Java Interface
XMLTree: an XML Toolbox For MATLAB/Octave
Denodo Virtual DataPort 8.0.2 API
Access Through JDBC — Virtual DataPort Developer Guide